A SIMILAR CONTENT RETRIEVAL METHOD FOR PODCAST EPISODES

Junta Mizuno

Nara Institute of Science and Technology
8916-5 Takayama Ikoma, Nara 630-0192, Japan
junta-m [at] is.naist.jp

ABSTRACT
Given podcasts (audio blogs) that are sets of speech files called episodes, this paper describes a method for retrieving episodes that have similar content. Although most previous retrieval methods were based on bibliographic information, tags, or users’ playback behaviors without considering spoken content, our method can compute content-based similarity based on speech recognition results of podcast episodes even if the recognition results include some errors. To overcome those errors, it converts intermediate speech-recognition results to a confusion network containing competitive candidates, and then computes the similarity by using keywords extracted from the network. Experimental results with episodes that have different word accuracy and content showed that keywords obtained from competitive candidates were useful in retrieving similar episodes. To show relevant episodes, our method will be incorporated into PodCastle, a public web service that provides full-text searching of podcasts on the basis of speech recognition.

Index Terms—Spoken document retrieval, Podcast, Speech recognition, Confusion network, Content-based similarity

1 Introduction
Since the amount of speech files on the Web is rapidly increasing, most previous methods for retrieving similar speech files on the basis of their bibliographic information are not useful enough because the bibliographic information is less informative than the spoken content recorded in speech files. Most popular speech files on the Web are called podcasts, which are often referred to as audio blogs. Each podcast consists of several speech/audio files (MP3 files) called episodes and a syndication feed (RSS) that includes metadata text information about episodes, such as the list of URLs of episode audio files and bibliographic information (titles, summaries, and published dates). By subscribing to the RSS feed of a podcast, a user can automatically download updated episodes of the podcast for playback on computers and portable media players. By using the RSS feed, however, it is difficult to find podcasts and episodes that are of interest. Just as various text retrieval services are essential for accessing text web pages, there is a growing need for spoken document retrieval services that can deal with spoken content of podcast episodes[1].

An example of such web services for podcasts is a full-text podcast search service called PodCastle[2][3], which enables a user to retrieve episodes on the basis of speech recognition results of spoken content. On this service, a user can access podcast episodes by finding ones that include a search term or simply by selecting one from a list of podcasts and episodes, which are listed, for example, according to the chronological order or the number of accesses. It is difficult, however, to use an episode itself as a search query — i.e., to find episodes that are similar to the search-query episode in terms of spoken content. Moreover, even if most video-sharing web services such as YouTube¹ have a function of recommending similar content, it is based on bibliographic information, tags, or past playback behaviors of users[4], which are often insufficient or not available for podcasts such as commentary or comedy.

Therefore, we propose a method that computes content-based similarity of podcast episodes on the basis of speech recognition results of those episodes. This method enables a user to find episodes that are similar to a query episode without using the RSS feed and bibliographic information. The biggest issue in computing such similarity is that speech recognition results are not perfect — i.e., they often include recognition errors. We tackle this issue by using not only the regular recognition result (1-best result) but also other competitive candidates in a confusion network that condenses huge intermediate speech recognition results (word lattice) into compact candidate representations. Even if the 1-best result is not correct, we can expect that the correct candidate is included in the confusion network. Since our method extracts keywords from this confusion network for each episode and computes content-based similarity based on those keywords, it can be robust to speech recognition errors.

2 Similar Episode Retrieval Method
To retrieve episodes on the basis of similarity of their content, we have to define a similarity measure between two episodes. A typical approach for computing similarity between two documents is to use a bag-of-words model that represents a text document as an unordered set of words appearing in its document, and then compute similarity between two models. However, if we use a large vocabulary continuous speech recognition (LVCSR) system to obtain a text

¹http://www.youtube.com
as the speech recognition result of each episode, such sim-
ple similarity based on bag-of-words models is not sufficient 
because the recognition result often includes recogni-
tion errors. In addition, the recognition result also includes 
unnecessary words/phrases, such as greetings, supportive response, 
and fillers that are not relevant to spoken content of an episode.

Our method therefore extracts a set of relevant and reliable 
keywords from the speech recognition result of each episode 
by using an original improved version of the traditional TF-
IDF (term frequency - inverse document frequency) weight. It 
then computes similarity between two sets of the keywords.

2.1 Keyword Extraction

Given a podcast episode, its keywords are extracted by using 
an improved version of the TF-IDF technique. The traditional 
TF-IDF technique [5] estimates the relative importance of each 
word, $w_i$, appearing in a text document (the recognition result 
of an episode) as a weight, TF-IDF, as follows:

$$\text{TF-IDF}_i = \text{TF}_i \times \text{IDF}_i, \quad (1)$$

$$\text{TF}_i = \frac{n_i}{\sum_k n_k}, \quad (2)$$

$$\text{IDF}_i = \log \frac{\#D}{\#d + 1}, \quad (3)$$

where $n_i$ is the number of occurrences of the word $w_i$ in this 
document/episode, $\#d$ is the number of documents/episodes 
that include the word $w_i$, and $\#D$ is the number of all doc-
uments/episodes. The TF-IDF weight is the product of the 
TF (term frequency) measuring the importance of the word $w_i$ 
in this document/episode and the IDF (inverse document fre-
cuency) measuring the uniqueness of the word $w_i$ in all doc-
uments/episodes.

To cope with speech recognition errors, we apply this TF-
IDF technique to a confusion network [6]. Note that the useful-
ness of the confusion network in retrieving similar speech files 
has already been tested and confirmed for a different purpose 
where a search query is not an episode itself (speech file) but 
a word or phrase (text term) [7][8]. Moreover, when a search 
query is a word or phrase, there have been other approaches 
such as [9][10][11]. On the other hand, our purpose is to test 
the usefulness of the confusion network by giving an episode as 
a search query and comparing it with every episode to find 
similar episodes on the basis of their keywords that are auto-
matically estimated from spoken content.

As shown in Figure 1, the confusion network is a com-
 pact and simplified network obtained by condensing a word 
lattice (intermediate speech recognition result). It was origi-
nally introduced in the context of a recent decoding algorithm, 
word error minimization, which minimizes the word error rate 
of recognition results rather than the sentence error rate [6]. 
Mangu et al. [6] reported that they obtained a compact confu-
ision network without any loss of correct words just by using 
5% of the number of links in the original huge lattice. Each 
candidate (word) of the confusion network has a posterior 
probability of speech recognition, which represents the rela-
tive reliability (i.e., confidence measure) of competitive candi-
dates. Since competitive candidates are ranked/sorted accord-
ing to the order of this probability, the top-ranked candidate 
corresponds to the regular recognition result (1-best result).

By using the confusion network, we can define an improved 
version of the TF-IDF weight, which considers not only the 
regular recognition result (1-best result) but also other com-
petitive candidates. In the following sections, we define the origi-
inal TF, IDF, and TF-IDF on the basis of the posterior proba-
bility of each candidate (word).

2.1.1 TF for confusion network

We define the TF of each word $w_i$ in the confusion network, 
named $\text{TFp}$, as a sum of all the posterior probabilities of words 
within each episode as follows:

$$\text{TFp}_i = \frac{\sum_k P_k(w_i)}{\sum_{w_i \in E} \sum_k P_k(w_i)}, \quad (4)$$

where $P_k(w_i)$ is the posterior probability of a word $w_i$ that 
is located at a position (segment) $k$ of the confusion network 
of an episode $E$. When there is a word with a low posterior 
probability (i.e., a word that was unlikely to be spoken in its 
episode), the $\text{TFp}$ of this word becomes low even if it appears 
in the confusion network.

2.1.2 IDF for confusion network

We compare two different versions of the IDF of each word 
$w_i$ in the confusion network. Both versions are the same as 
the traditional IDF except for the way of counting (enumerat-
ing) word occurrence. The first IDF, named $\text{IDFn}$, is calculated 
by enumerating each occurrence of the word $w_i$ in the confu-
sion network as one word occurrence (1.0), even if its posterior 
probability is very low. The second IDF, named $\text{IDFp}$, is calcu-
lated by regarding the highest posterior probability of the word 
$w_i$ within each episode as its word occurrence for this episode:

$$\text{IDFp}_i = \log \frac{\#D}{\sum_{w_i \in E} \max_k(P(w_i)) + 1}, \quad (5)$$

where $\max_k(P(w_i))$ denotes the highest posterior probability 
of the word $w_i$ in an episode $E$.

2.1.3 TF-IDF for confusion network

By denoting the traditional/baseline TF by $\text{TFb}$ and the tra-
ditional/baseline IDF by $\text{IDFb}$, we define the following three 
versions of the TF-IDF of each word $w_i$:

$\text{TFb-IDFb}$ This is the baseline TF-IDF that uses only the 1-
best speech recognition result without using the confusion 
network and posterior probabilities.

$\text{TFp-IDFn}$ The posterior probability in the confusion network 
is used for the $\text{TFp}$, but not for the $\text{IDFn}$. 

![Fig. 1: Example of confusion network: boldfaced candidates were correct.](image-url)
Evaluation metric 1

\[ \text{Reciprocal Rank (MRR)} = \frac{1}{\sum_{k=1}^{N} \frac{1}{r_{k}}} \]

where \( N \) is the number of episodes to be evaluated (\( N = 10 \)), and \( s_k \) is the score (0 ~ 3) of the \( k \)-th ranked episode. Note that \( \frac{1}{r_k} \) is a weight according to the rank \( k \): the lower the rank, the smaller the weight. If a user listens to all the top 10 retrieved episodes, this evaluation is important.

The evaluation result by metric 1 is shown in Table 2 and the result by metric 2 is shown in Figure 2. Metric 1 takes a value from 1 to 10: the smaller, the better. Metric 2 takes a value from 0 to 1: the higher, the better.

### 3.4 Discussion

Appropriate keywords were extracted in most cases: for example, we obtained keywords such as “post office,” “bank transfer,” “system,” “computer,” “ubiquitous,” and “payment” for colm2 about bank payment, and keywords such as “kilocalorie,” “dieting,” and “weight” for talk1 about dieting.

We found that the results with TFp-IDFp were good in general except for colm2 and talk2. Since the word accuracy of these two episodes was too low as shown in Table 1, the quality of the confusion network was not good enough to improve the performance. The word accuracy of talk1 was also low, but a unique relevant keyword, “dieting,” was extracted and utilized in the similarity computation, and the performance was good.
The difference between TFp-IDFn and TFp-IDFp is the IDF computation, and we investigated cases where the TFp-IDFp was better. In theory, the IDFn and IDFn of keywords with high posterior probabilities can have almost the same values, but the IDFn of keywords with low posterior probabilities is always lower than their IDFn. When a correct keyword had a low posterior probability but appeared often in an episode, the TFp-IDFp was better than the TFp-IDFn because its IDFn was higher than its IDFn. In addition, if correct keywords have low TFp-IDFn weights in this way, incorrect keywords are likely to appear in the top 100 keywords, resulting in worse similarity computation.

4 Conclusion

We have described a similar episode retrieval method that is based on spoken content of podcast episodes. Our method uses an improved version of the TF-IDF weight, TFp-IDFp, which considers a confusion network that has competitive candidates and the posterior probability (reliability) of the candidates. We found that it was effective when the word accuracy was not too low.

In the future, it will become important to automatically optimize the number of keywords used in the similarity computation, especially for very short or long episodes. It can be changed, for example, according the temporal duration of each episode: longer episodes could have more keywords. We plan to apply our proposed method to a larger data set and evaluate it by using inter-rater agreement such as kappa measure among several observers. In addition, we plan to automatically switch/merge the three TF-IDF versions according to the word accuracy predicted on the basis of posterior probabilities of the confusion network. Future work will also include automatic clustering of episodes on the basis of the proposed similarity measure and integration of our method with the full-text podcast search service PodCastle [2].

Acknowledgements

This work was partially supported by Grant-in-Aid for Scientific Research (B) (KAKENHI 19300065).

5 References


